2. Why Use Node.js?

Node.js is a powerful platform that brings **high performance** and **efficiency** to web applications. Here's why developers prefer it:

2.1 High Performance

- Uses the **V8 JavaScript engine**, which compiles JavaScript to machine code for faster execution.
- Asynchronous, non-blocking execution means better resource utilization.

2.2 Single Programming Language

- Allows **JavaScript for both frontend & backend**, reducing complexity.
- Developers don't need to learn multiple languages for full-stack development.

2.3 Non-Blocking I/O Model

- Uses **event-driven architecture** for handling multiple requests simultaneously.
- Unlike traditional web servers (like PHP or ASP.NET), it doesn't block the execution thread.

2.4 Scalable and Lightweight

- Perfect for real-time applications like **chat apps, streaming services, and IoT applications**.
- Works well for microservices architecture due to its lightweight nature.

2.5 Rich Ecosystem (NPM)

> Node.js has **NPM (Node Package Manager)**, with over **2 million open-source packages** for rapid development.

---

3. How Node.js Works? (Architecture)

Node.js follows an **event-driven, non-blocking I/O model** powered by the **libuv** library.

3.1 Event-Driven Architecture

Node.js runs a **single-threaded event loop** that listens for events and delegates tasks to worker threads when needed.

3.2 Non-Blocking I/O

- When an I/O request (like database calls, file reading, or API requests) is made, Node.js **doesn't wait** for the response.
- Instead, it moves on to the next task and processes the response asynchronously when it's ready.

4. Installing Node.js

To use Node.js, follow these steps:

4.1 Download & Install

1. Go to the official Node.js website.
2. Download the **LTS (Long-Term Support)** version.
3. Install it by following the instructions for your OS (Windows, Mac, Linux).

4.2 Verify Installation

Open a terminal and run:

```
node -v
```

If Node.js is installed, it will display the version number.

To check **NPM (Node Package Manager)** version:

```
npm -v
```

5. Creating Your First Node.js Application

5.1 Hello World in Node.js

Create a file `app.js` and add the following code:

```
// Importing the HTTP module
const http = require('http');

// Creating a server
const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, World! Welcome to Node.js!');
});

// Running the server on port 3000
server.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

Run the file in terminal using:

```
node app.js
```

Now, open a browser and go to **http://localhost:3000/**. You will see **"Hello, World! Welcome to Node.js!"**

---

6. Node.js Core Modules

Node.js comes with several **built-in modules** to perform different tasks.

Module	Description
`http`	Create web servers
`fs`	File system operations (read/write files)
`path`	Work with file paths
`os`	Get OS-related information
`events`	Handle events in an application
`crypto`	Perform encryption and hashing

Example: Reading a file using the `fs` module

```
const fs = require('fs');

fs.readFile('test.txt', 'utf8', (err, data) => {
    if (err) {
        console.error(err);
        return;
    }
    console.log(data);
});
```

---

7. Node.js with Express.js

Express.js is the most popular framework for Node.js, used to build web applications and RESTful APIs.

Installing Express.js

```
npm install express
```

Basic Express.js Server

```
const express = require('express');
const app = express();

// Define a simple route
app.get('/', (req, res) => {
    res.send('Hello, Express.js!');
});

// Start the server
app.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

---

8. Real-World Applications of Node.js

Node.js is used in various industries for different applications.

Application Type	Examples
Real-time Chat Apps	WhatsApp Web, Discord, Slack
Streaming Services	Netflix, YouTube, Twitch
E-commerce Platforms	eBay, Walmart, AliExpress
IoT Applications	Smart Home Automation

---

9. Advantages & Disadvantages of Node.js

Advantages

✅ **Fast Performance** – Uses V8 engine & asynchronous execution.

✅ **Scalability** – Perfect for handling high traffic applications.

✅ **Full-Stack JavaScript** – One language for frontend & backend.

✅ **Large Ecosystem** – Rich library support via NPM.

✅ **Community Support** – Huge open-source community.

Disadvantages

❌ **Single-Threaded Limitation** – Not ideal for CPU-intensive tasks.

❌ **Callback Hell** – Too many nested callbacks can make code hard to read.